Event Structures for Mixed Choice

Marc de Visme
Univ Lyon, EnsL, UCBL, CNRS, LIP, F-69342, LYON Cedex 07, France

Abstract

In the context of models with mixed nondeterministic and probabilistic choice, we present a concurrent model based on partial orders, more precisely Winskel's event structures. We study its relationship with the interleaving-based model of Segala's probabilistic automata. Lastly, we use this model to give a truly concurrent semantics to an extension of CCS with probabilistic choice, and relate this concurrent semantics to the usual interleaving semantics, thus generalising existing results on CCS, event structures and labelled transition systems.

2012 ACM Subject Classification Theory of computation → Parallel computing models

Keywords and phrases probability, nondeterminism, concurrency, event structures, CCS

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2019.11

Related Version Full version at `https://hal.archives-ouvertes.fr/hal-02167420`.

Funding *Marc de Visme*: Supported by Labex MiLyon (ANR-10-LABX-0070) of Université de Lyon, within the program "Investissements d'Avenir" (ANR-11-IDEX-0007), operated by the French National Research Agency (ANR).

1 Introduction

We study models of *mixed choice* [7], i.e. models representing both probabilistic choice and nondeterministic choice. The need for such models arises with systems where unconstrained nondeterministic behaviours coexist with quantified and controlled nondeterministic behaviours; for example, parallel threads using random number generators (hence probabilistic choices) while operating on a shared memory (hence nondeterministic races).

While many different models have been developed through the years, Segala's probabilistic automata [13, 14] are a widely used model, both general and practical. They are automata where transitions are from states to probability distributions on states, hence modelling an alternation between a nondeterministic choice (the choice of the transition) and a probabilistic choice (the probability distribution). It is an *interleaving model*, as it represents "A and B occur in parallel" by "A then B or B then A". In this paper, we are interested in models that are not interleaving, and represent "A and B occur in parallel" by "A and B are causally unrelated" instead. Those models are called *truly concurrent models*, and are particularly useful to study races, concurrency, and causality. We specifically focus on truly concurrent models based on partial orders, more precisely Winskel's event structures [9, 11, 17]. We present here *mixed probabilisitic event structures*, which are event structures enriched to model mixed nondeterministic and probabilistic choice. This work is in continuation of works on event structure models for languages with effects: parallelism [17], probabilities [5, 16], quantum effects [6], shared weak memory [4], ...

In order to ensure that mixed probabilistic event structures are an adequate model for mixed choice, we show how to relate them to the existing model of Segala automata. Indeed, a mixed probabilistic event structure can be unfolded to a (tree-like) Segala automaton through a *sequentialisation* procedure, similar to the unfolding of a partial order into a tree. This sequentialisation procedure is well-behaved; rather than listing all the ad-hoc

properties it satisfies, we express mixed probabilisitic event structures as a category, such that sequentialisation forms an adjunction, from which those properties can be deduced. Sections 3 and 4 are dedicated to the development of this new model for mixed choice.

We apply this model to give a semantics to a language featuring parallelism, nondeterminism and probabilistic choice. Namely, we choose an extension of CCS [10] with probabilistic choice [2]. Extensions of our work to more complex languages, such as the probabilistic π-calculus, should be possible using methods similar to [16]. In that paper, Varacca and Yoshida give a concurrent semantics using event structures to a "deterministic" probabilistic π-calculus, i.e. they forbid processes with a nondeterministic behaviour, like $(a|a|\bar{a})$ where two inputs race to interact with an output. We do not have those restrictions, and fully support both nondeterministic and probabilistic behaviours.

Using partial-order based models to represent parallel processes is not a new idea. Indeed, the relationship between CCS and event structures is well-studied [17], and recalled in detail in Section 2. The core result of this relationship is the factorisation theorem (Theorem 9), which states that if one considers the event structure representing the concurrent semantics of a process, and unfolds it into a labelled tree, then the result would match the interleaving semantics of the process. It follows that the concurrent semantics is *sound* w.r.t. the interleaving semantics. In Section 4, we lift the known relations between CCS, event structures, and labelled trees, to relations between Probabilistic CCS, mixed probabilisitic event structures, and (tree-like) Segala automata; concluding with a factorisation theorem (Theorem 26).

2 Preliminaries

The process algebra CCS [10] is used to represent concurrent processes communicating with each other through channels. A process P can evolve into a process Q by performing an action. The graph having processes as nodes and transitions labelled by actions as edges is known as the Labelled Transition System (LTS) of the process language. From this graph and any process P, one can work with the (possibly infinite) labelled tree obtained by unfolding the graph starting by the node P. We may express the semantics of a process P in terms of that unfolded tree, the approach we take in this paper.

The LTS of processes yield an *interleaving semantics*, since they do not distinguish the process $(a|b)$ that perform a and b in parallel from the process $(a.b \between b.a)$ – where $\between$ represents nondeterministic choice – which can perform a and b in any sequential order. Semantics that do distinguish between the two actions in parallel and the two actions in any sequential order are called *truly concurrent semantics*. In this paper, we will consider the usual truly concurrent semantics of CCS: event structures [9, 11, 17], which are partial orders with a notion of conflict (Definition 1).

The semantics of CCS in labelled trees and in event structures are both quite straightforward, except for the parallel composition $(P|Q)$, which is syntactically complex to compute. It is often practical to use a more abstract approach than syntactically computing $(P|Q)$, and remark that the parallel composition of CCS arises in both semantics [17] as a categorical product $\times_\star$ (for a suitable notion of maps) followed by a restriction.

As stated before, we can recover the interleaving semantics from the truly concurrent semantics. This means that we can unfold event structures as labelled trees, while preserving the interpretation of CCS processes. This unfolding consists in finding the best labelled tree to approximate the behaviour of a given event structure.

In the literature of models of concurrency, following Winskel [12, 17], such unfoldings are often expressed through a categorical *coreflection* – a special case of adjunction. The unfolding of an event structure into a labelled tree, named *sequentialisation*, is the right

adjoint of this adjunction, ensuring that the sequentialisation $Seq(E)$ of an event structure E gives its best approximation in terms of labelled trees. In particular the sequentialisation of a product $Seq(E \times_\star F)$ is the product of the sequentialisations $Seq(E) \times_\star Seq(F)$; hence the sequentialisation preserves the interpretation of the parallel composition. This adjunction is a *coreflection*, meaning that the left adjoint, named *inclusion*, is such that the composition $(Seq \circ \hookrightarrow)$ is an isomorphism – in other words, labelled trees can be regarded as a particular case of event structures.

In this section, we first recall the definition of event structures and their cartesian category. Then, we recall the syntax of the process algebra CCS, its interleaving semantics using labelled trees, and its truly concurrent semantics using event structures. Finally, we present the extension of CCS with probabilistic choice [2], and its interleaving semantics.

2.1 The Category of Event Structures

A (prime) event structure [18] is a representation of computational events of a concurrent system. Its events are related by a partial order representing causal dependency: if $a \leq b$ then b can only occur if a has occurred beforehand; and a conflict relation representing incompatibility: if $a \# b$ then each of a and b can only occur if the other does not.

▶ **Definition 1.** *An* event structure E *is a triple* $E = (|E|, \leq_E, \#_E)$ *where:*

- $|E|$ *is a set whose elements are called events.*
- $\leq_E$ *is a (partial) order, called causality, such that* $\{b \mid b \leq_E a\}$ *is finite for all* $a \in |E|$.
- $\#_E$ *is a symmetric irreflexive binary*[1] *relation, called conflict.*
- $\forall a, b, c \in |E|$, *if* $a \#_E b \leq_E c$ *then* $a \#_E c$.

In such a structure, we want to describe the set of states in which the system under study can exist. We define the set of (finite) configurations of E, written $\mathcal{C}(E)$, as the set of reachable finite sets of events, in other words:

$$x \in \mathcal{C}(E) \iff x \in \mathcal{P}_{\text{fin}}(|E|) \text{ and } \forall a \in x, \forall b \in |E|, \begin{cases} b \#_E a \implies b \notin x & \text{and} \\ b \leq_E a \implies b \in x \end{cases}$$

We say that a configuration x *enables* an event $a \notin x$, and we write $x \overset{a}{\text{—}\!\subset}$, if $x \cup \{a\}$ is a configuration. There are two canonical configurations associated to an event $a \in |E|$: the minimal configuration containing it $[a] := \{b \mid b \leq_E a\}$, and the minimal configuration enabling it $[a) := [a] \backslash \{a\}$.

The causality $\leq_E$ and the conflict $\#_E$ contain a lot of redundant information: if you know that $a \#_E b$ and $b \leq_E c$, then the definition of event structures enforces $a \#_E c$. When representing event structures, we want a concise representation, so instead of $\leq_E$ and $\#_E$, we use immediate causality $\rightarrowtriangle_E$ (represented by arrows) and minimal conflict $\sim\!\!\!\sim_E$ (represented by wiggly lines), defined as follows:

$$a \rightarrowtriangle_E b \iff \begin{cases} a <_E b \\ \forall a \leq_E c \leq_E b, c \in \{a, b\} \end{cases} \qquad a \sim\!\!\!\sim_E b \iff \begin{cases} a \#_E b \\ \forall a' <_E a, \neg(a' \#_E b) \\ \forall b' <_E b, \neg(a \#_E b') \end{cases}$$

We can always recover $\#_E$ and $\leq_E$ from $\rightarrowtriangle_E$ and $\sim\!\!\!\sim_E$. For example, in Figure 1, we deduce $c \#_E e$ and $a \leq_E e$ from the minimal conflict and immediate causality.

[1] We choose to use binary conflicts for pedagogical reasons. Our work can be extended to non-binary conflicts as in [19].

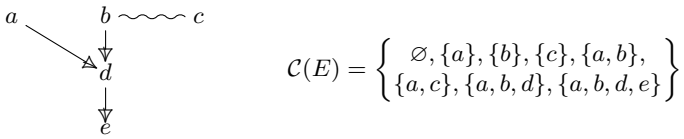


Figure 1 An event structure E.

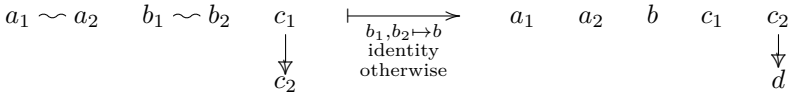


Figure 2 A total map of event structures.

Figure 3 Concurrent system, and its interleaving counterpart.

There is a notion of (partial or total) maps of event structures, altogether forming a category. Formally, a (partial or total) map f from E to E' is a (partial or total) function $f : |E| \rightharpoonup |E'|$ such that:
configuration preserving for all $x \in \mathcal{C}(E)$ we have $f(x) \in \mathcal{C}(E')$.
local injectivity for all a, b distinct in $x \in \mathcal{C}(E)$, if f is defined on both then $f(a) \neq f(b)$.

Note that total maps can be interpreted as a form of simulation: if $x \in \mathcal{C}(E)$ enables an event $e \notin x$, then $f(x) \in \mathcal{C}(E')$ enables $f(e) \notin f(x)$. In the example of Figure 2, the map is total. It merges b_1 and b_2 into a single event b, which is allowed since they are in conflict, and it does not contain the event d in its image, which is allowed because d is not causally required by any event in the image of the map.

With this notion of maps, we recall the induced notion of isomorphism: $E \cong E'$ if and only if there is a total bijective map $f : E \to E'$ with f^{-1} also a map of event structures. In other words, two event structures are isomorphic if and only if they are the same up to renaming of the events.

▶ **Proposition 2.** *Event structures and (partial) maps of event structures form a category, written* $\mathsf{ES}_\star$*. We write* $\varnothing$ *for the empty event structure, which is a terminal object*[2]*.* $\mathsf{ES}_\star$ *has a subcategory* ES *of event structures with total maps of event structures.*

Event structures are used to represent causality and independence, but they can also be used to represent interleavings. However, concurrent systems and sequential systems simulating concurrency through interleaving will be represented in drastically different ways in event structures. Figure 3 shows a concurrent system able to perform two events a and b in parallel, and its interleaving counterpart where a and b can both happen in any order but not simultaneously. We can characterise event structures that are fully interleaved, in other words sequential event structures, as follows:

[2] i.e. for every object A, there exists a unique map from A to $\varnothing$.

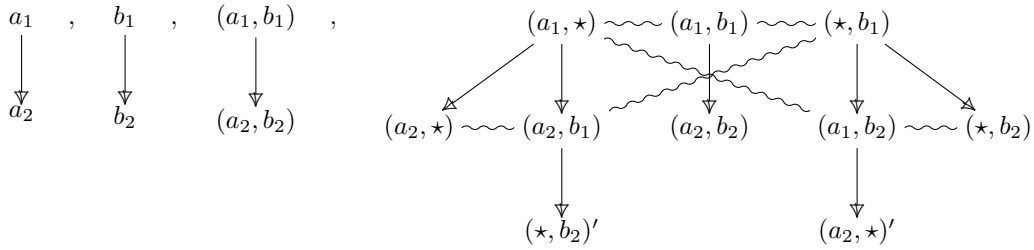


Figure 4 The event structure A, B, $A \times B$ and $A \times_\star B$.

▶ **Definition 3.** *An event structure E is sequential if it satisfies one of the following equivalent properties:*

- *E is forest-shaped, with branches being in conflict with each other.*
- *For every $a, b \in x \in \mathcal{C}(E)$, either $a \leq_E b$ or $b \leq_E a$.*
- *There exists a (necessarily unique) total map from E to $\mathcal{N}$, where $\mathcal{N} = (\mathbb{N}, \leq, \varnothing)$ is the event structure with an infinite succession of events.*
- *E is isomorphic to $E \times \mathcal{N}$, where $\times$ is the synchronous product defined in Section 2.2.*

The first characterisation means that trees are in a one-to-one correspondence with sequential event structures. Through this correspondence, transitions correspond to events. For the remainder of this paper, we use sequential event structures to represent trees.

▶ **Proposition 4.** *Sequential event structures and (partial) maps of event structures form a subcategory of $\mathsf{ES}_\star$, written $\mathsf{Seq\text{-}ES}_\star$. It has a subcategory $\mathsf{Seq\text{-}ES}$ of sequential event structures with total maps of event structures, with $\mathcal{N}$ for terminal object.*

2.2 The Categorical Product

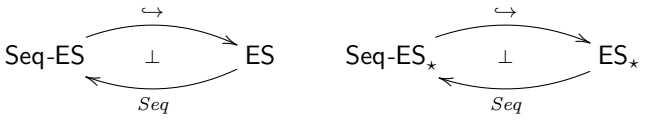
In this paper, we exclusively rely on the universal property of categorical products, without giving any concrete definition for such products. We write $\times_\star$ for the product in $\mathsf{ES}_\star$ [17], called *asynchronous product*, and $\times$ for the product in ES [17], called *synchronous product*.

Informally speaking, both $A \times_\star B$ and $A \times B$ explore all the ways to pair up the events of A with the events of B while respecting causal dependencies and conflicts. The synchronous product expects all the events to be paired, while the asynchronous product allows events to remain unpaired, see Figure 4. Note that events of the products are not uniquely determined by their projections: in Figure 4 both $(\star, b_2)$ and $(\star, b_2)'$ have the same projections.

▶ **Proposition 5.** *$\mathsf{ES}_\star$ is a cartesian category, with $\times_\star$ as a product and $\varnothing$ as a unit. ES has a product $\times$, and its subcategory $\mathsf{Seq\text{-}ES}$ is a cartesian category with $\mathcal{N}$ as unit.*

Using the synchronous product $\times$, we can simply define the sequentialisation as $Seq(E) := E \times \mathcal{N}$. It is always a sequential event structure. The asynchronous product $\times_\star$ will later be used to define the semantics of the parallel composition of CCS.

▶ **Theorem 6.** *Seq is a functor from $\mathsf{ES}_\star$ (and ES) to $\mathsf{Seq\text{-}ES}_\star$ (and $\mathsf{Seq\text{-}ES}$), which is right adjoint to the inclusion functor, forming coreflections [17].*



Since right adjoints preserve categorical limits, we have that $Seq(A \times_\star B) \cong Seq(A) \times_\star Seq(B)$. Since the restriction (Definition 8) is also preserved by Seq, it means that when interpreting the parallel composition of CCS using the asynchronous product in $\mathsf{ES}_\star$, then sequentialising, we get the same result as when interpreting the parallel composition directly as the asynchronous product in $\mathsf{Seq\text{-}ES}_\star$ (i.e. trees).

2.3 Semantics of CCS

We consider the process algebra CCS [1, 3, 10] over a set of channels $Chan$. A channel $a \in Chan$ can be used by processes as an input a or as an output $\bar{a}$. For convenience, we assume that $\bar{\bar{a}} = a$. On top of these external actions, processes can also perform internal actions, written τ, and we write $\mathbb{A} = Chan \cup \{\bar{a} \mid a \in Chan\} \cup \{\tau\}$ for the set of all actions.

▶ **Definition 7.** *The set $\mathbb{P}$ of processes is defined by the following syntax:*
$P ::= \mathbf{0} \mid (P_1|P_2) \mid (P_1 \between P_2) \mid X \mid \mu X.P \mid a.P \mid \bar{a}.P \mid \tau.P \mid \nu a.P$ *(for $a \in Chan$)*
where the constructs are empty process, parallel composition, nondeterministic choice, process variable, process recursion, input prefix, output prefix and τ prefix, and restriction, respectively. We only consider closed processes, in which every variable X is bound by a recursion μX.

Figure 5a describes the interleaving semantics of the language. The states of the LTS reachable from a process P can be unfolded into a (potentially infinite) labelled tree represented using a (potentially infinite) sequential event structure, written $[\![P]\!]_{\text{is}}$, with labels in $\mathbb{A}$.

▶ **Definition 8.** *An event structure with labels in $\mathcal{L}$ is an event structure E together with a total function $\ell_E : |E| \to \mathcal{L}$. For $L \subseteq \mathcal{L}$, we define $E\backslash L$ as the restriction of E to the set of events $\{e \mid \forall e' \leq e, \ell_E(e') \notin L\}$. In other words, we remove every event with a label in L, and every event causally dependent on it. Maps of labelled event structures are required to preserve labels.*

We define the labels on $Seq(E)$ from the labels on E as $\ell_{Seq(E)}(e) := \ell_E(\pi_1(e))$. We define the labels of $A \times_\star B$ using the following synchronisation operation $\bullet : \mathbb{A} \cup \{\star\} \times \mathbb{A} \cup \{\star\} \to \mathbb{A} \cup \{\mathbf{0}\}$, where $\star$ stands for "undefined": $\ell_{A\times_\star B}(e) = \ell_A(\pi_1^\star(e)) \bullet \ell_B(\pi_2^\star(e))$

$a \bullet \bar{a} = \tau$ and $\bar{a} \bullet a = \tau$ for $a \in Chan$ $\qquad$ $\alpha \bullet \star = \alpha$ and $\star \bullet \alpha = \alpha$ for $\alpha \in \mathbb{A}$ $\qquad$ $\alpha \bullet \beta = \mathbf{0}$ otherwise

Figure 5b describes the concurrent semantics of the language. For any process P, we write $[\![P]\!]_{\text{cs}}$ for the (potentially non-sequential, potentially infinite) event structure with labels in $\mathbb{A}$ representing P. See Figure 6 for an example. On top of the previous operations, we need some additional operations on labelled event structures:

- To represent nondeterministic choice, we need to put two event structures in parallel, while allowing only one of them to be used. For A and B two event structures, we define $A \# B$ as the event structure with $|A| \uplus |B|$ as events, with the same order, conflict and labels as in A and in B, but with every event of A being in conflict with every event of B.
- To represent prefixes, we need to create a new event. We write $\downarrow_{e:a} E$ for the event structure with events $|E| \uplus \{e\}$, with e labelled by a being the minimal event for $\leq_{\downarrow_{e:a} E}$, everything else being the same as in E.
- To represent process recursion, we use a least fixpoint. The order used for that least fixpoint is given by "substructure maps", i.e. total maps that are an inclusion on events, and preserve and reflect order and conflicts.

As claimed before, we can recover the interleaving semantics from the truly concurrent one. The proof of this theorem relies on the fact that Seq forms a coreflection.

▶ **Theorem 9** (Factorisation [17])**.** *For any process P of CCS, we have $[\![P]\!]_{is} \cong Seq([\![P]\!]_{cs})$.*

$$\frac{}{a.P \xrightarrow{a} P} \qquad \frac{P \xrightarrow{a} P'}{(P \between Q) \xrightarrow{a} P'} \qquad \frac{P \xrightarrow{a} P'}{(Q \between P) \xrightarrow{a} P'}$$

$$\frac{a \notin \{c, \bar{c}\} \quad P \xrightarrow{a} P'}{\nu c.P \xrightarrow{a} \nu c.P'} \qquad \frac{P \xrightarrow{a} P'}{(P|Q) \xrightarrow{a} (P'|Q)} \qquad \frac{P \xrightarrow{a} P'}{(Q|P) \xrightarrow{a} (Q|P')}$$

$$\frac{P \xrightarrow{c} P' \quad Q \xrightarrow{\bar{c}} Q'}{(P|Q) \xrightarrow{\tau} (P'|Q')} \qquad \frac{P\{X \leftarrow \mu X.P\} \xrightarrow{a} Q}{\mu X.P \xrightarrow{a} Q}$$

(a) Interleaving semantics, using LTS.

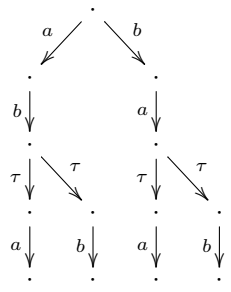
$$\begin{aligned}
[\![\mathbf{0}]\!]_{\text{cs}} &= \varnothing \\
[\![(P_1|P_2)]\!]_{\text{cs}} &= [\![P_1]\!]_{\text{cs}} \times_\star [\![P_2]\!]_{\text{cs}} \backslash \{\mathbf{0}\} \\
[\![(P_1 \between P_2)]\!]_{\text{cs}} &= [\![P_1]\!]_{\text{cs}} \# [\![P_2]\!]_{\text{cs}} \\
[\![a.P]\!]_{\text{cs}} &= \downarrow_{P:a} [\![P]\!]_{\text{cs}} \\
[\![\nu c.P]\!]_{\text{cs}} &= [\![P]\!]_{\text{cs}} \backslash \{c, \bar{c}\} \\
[\![\mu X.P]\!]_{\text{cs}} &= [\![P\{X \leftarrow \mu X.P\}]\!]_{\text{cs}}
\end{aligned}$$

(b) Concurrent semantics, using event structures.

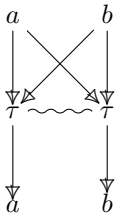
Figure 5 Semantics of CCS, with $a \in \mathbb{A}$ and $c \in \mathit{Chan}$.

$\nu c.\,(a.\,(c.a \between c.b) \mid b.\bar{c})$

(a) A CCS process.



(b) Its interleaving semantics: a labelled tree.



(c) Its concurrent semantics: a labelled event structure.

Figure 6

2.4 Probabilistic CCS

The goal of this paper is to extend existing results on CCS to Probabilistic CCS, which is CCS enriched with a probabilistic choice[3] $\sum_{i \in I} p_i \cdot a_i.P_i$, with I a possibly infinite set, $\forall i \in I, 0 < p_i \leq 1$, $a_i \in \mathbb{A}$, and $\sum_{i \in I} p_i \leq 1$. We also assume the pairs (a_i, P_i) to be pairwise distinct. Notice that we allow sub-probabilistic sums like $\frac{1}{2} \cdot a.P$, but we forbid unguarded sums like $\frac{1}{2} \cdot (P_1|P_2) + \frac{1}{2} \cdot Q$. This guard restriction is similar to that found in the probabilistic π-calculus [8, 16]. The interleaving semantics we present here uses Segala automata [13, 14], and relies on this absence of unguarded sums to be well-defined. In this semantics, reductions are interpreted by an alternation of probabilistic choices and nondeterministic choices. Mathematically, the reduction $\to$ is a subset of $\mathbb{P} \times \mathcal{D}(\mathbb{A} \times \mathbb{P})$, where $\mathcal{D}(U)$ is the set of discrete sub-probability distributions over U. So for every process P, there is first a nondeterministic choice of $P \to S$, and then a probabilistic choice inside $S = \{(p_i, a_i, P_i) \mid i \in I\}$ of the action a_i and the reduced process P_i. Figure 7 describes this interleaving semantics.

3 Mixed Event Structures

We now develop the main contribution of this paper: an event structure model able to represent mixed internal and external choices, that we will use in the last section to give a concurrent semantics to the mixed probabilistic and nondeterministic choices of PCCS.

[3] Some papers [2] use a less general sum where the a_i are assumed to be equal.

$$\frac{a \in \mathbb{A}}{a.P \to \{(1, a, P)\}} \qquad \frac{}{\sum_{i\in I} p_i \cdot a_i.P_i \to \{(p_i, a_i, P_i) \mid i \in I\}} \qquad \frac{P \to S}{P \oplus Q \to S} \qquad \frac{Q \to S}{P \oplus Q \to S}$$

$$\frac{P\{X \leftarrow \mu X.P\} \to S}{\mu X.P \to S} \qquad \frac{P \to \{(p_i, a_i, P_i) \mid i \in I\}}{\nu c.P \to \{(p_i, a_i, P_i) \mid i \in I, a_i \notin \{c, \bar{c}\}\}} \qquad \frac{P \to \{(p_i, a_i, P_i) \mid i \in I\}}{P|Q \to \{(p_i, a_i, P_i|Q) \mid i \in I\}}$$

$$\frac{P \to \{(p_i, a_i, P_i) \mid i \in I\} \qquad Q \to \{(q_j, b_j, Q_j) \mid j \in J\}}{P|Q \to \{(p_i q_j, \tau, P_i|Q_j) \mid i \in I, j \in J, \bar{a_i} = b_j\}} \qquad \frac{Q \to \{(q_j, b_j, Q_j) \mid j \in J\}}{P|Q \to \{(q_j, b_j, P|Q_j) \mid j \in J\}}$$

Figure 7 Interleaving semantics of PCCS, using Segala automata.

3.1 Definition

When trying to represent processes of PCCS in event structures, one could think of simply using the existing probabilistic event structures [20], which are event structures E together with a valuation $v_E : \mathcal{C}(E) \to (0, 1]$ satisfying some well-chosen properties (Definition 21). However, looking at the two processes $(\frac{1}{2}a + \frac{1}{2}b)$ and $\big((\frac{1}{2}a) \oplus (\frac{1}{2}b)\big)$, we remark that:

- They have only two different computational events: "receiving on a" and "receiving on b"; which means that natural representations using event structures will only have two events.
- Those two computational events cannot occur together, so natural representations using event structures have those two events in conflict.
- Those two computational events are associated with probability half, which means that the valuation is necessarily $v(\{a\}) = \frac{1}{2} = v(\{b\})$

In other words, if we try to represent them using probabilistic event structures, both of them will be represented by the same structure. But they have very different interleaving semantics in Segala automata, so it would not be a sound representation. A similar example using full probabilistic distributions is $\big(\frac{1}{2}a + \frac{1}{2}b) \oplus (\frac{1}{2}c + \frac{1}{2}d)\big)$ and $\big(\frac{1}{2}a + \frac{1}{2}c) \oplus (\frac{1}{2}b + \frac{1}{2}d)\big)$.

In order to distinguish these two processes, the solution we propose is to have two kinds of conflicts: an *external conflict*, used for the nondeterministic choice $\oplus$, and an *internal conflict*, used for the probabilistic choice $+$. In this section, we explore in more detail event structures with two kinds of conflicts, named *mixed event structures*.

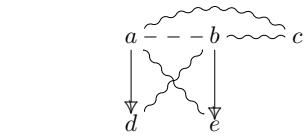
We will keep using the notation $\#$ for the union of both conflicts, and we will use the notation $\square$ for internal conflicts and $\lrcorner\!\!\urcorner$ for external conflicts. Since we have two kinds of conflicts, we will have two kinds of "configurations": the usual set of configurations, computed from $\#$, inside which no conflicts are tolerated, and the set of *worlds* inside which internal conflicts $\square$ are accepted. These considerations give rise to the following definition.

▶ **Definition 10.** *A* mixed event structure *(mes) E is a quadruple $(|E|, \leq_E, \#_E, \lrcorner\!\!\urcorner_E)$ where*

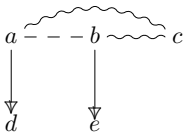
- *$\mathcal{U}(E) = (|E|, \leq_E, \#_E)$ is an event structure. We write $\mathcal{C}(E) = \mathcal{C}(\mathcal{U}(E))$ for the set of* configurations*. We define $\rightsquigarrow_E, \twoheadrightarrow_E, [e], [e)$ as previously.*
- *$(|E|, \leq_E, \lrcorner\!\!\urcorner_E)$ is an event structure. We write $\mathcal{W}(E) = \mathcal{C}(|E|, \leq_E, \lrcorner\!\!\urcorner_E)$ for the set of* worlds*.*
- *$\lrcorner\!\!\urcorner_E \subseteq \#_E$, or equivalently $\mathcal{C}(E) \subseteq \mathcal{W}(E)$.*

We define the internal conflict $\square_E$ as $(\#_E \backslash \lrcorner\!\!\urcorner_E)$.

The internal conflict $\square_E$ is a symmetric irreflexive relation, but $(|E|, \leq_E, \square_E)$ may not be an event structure. In fact, we will later (Definition 16) consider the special case $\square_E \subseteq \rightsquigarrow_E$. The operation $\mathcal{U}$ turns out to be the right adjoint in a coreflection (Theorem 13) between mes and event structures. Graphically, we use dashes to represent minimal internal conflict (i.e. $\rightsquigarrow_E \cap \square_E$), wiggly lines to represent the minimal conflict of $(|E|, \leq_E, \lrcorner\!\!\urcorner_E)$, and arrows to represent $\twoheadrightarrow_E$. Those are enough to characterise all the conflicts. For example, in Figure 8, we can deduce $c \lrcorner\!\!\urcorner e$ in both E_1 and E_2, in E_1 we also have $d \lrcorner\!\!\urcorner e$ while in E_2 we have $d \square e$.



$\mathcal{C}(E_1) = \{\varnothing, \{a\}, \{b\}, \{c\}, \{a,d\}, \{b,e\}\}$
$\mathcal{W}(E_1) = \mathcal{C}(E_1) \cup \{\{a,b\}\}$



$\mathcal{C}(E_2) = \mathcal{C}(E_1)$
$\mathcal{W}(E_2) = \mathcal{W}(E_1) \cup \{\{a,b,d\}, \{a,b,e\}, \{a,b,d,e\}\}$

Figure 8 Mixed event structures E_1 and E_2.

Figure 9 Image by the sequentialisation of the mes A and B.

Our goal is to extend all the operations previously defined on event structures. So we want mes to have (a)synchronous products, and a sequentialisation *Seq* functor forming a coreflection. In particular, it means sequentialising a mes which is already sequential (Definition 16) should be an isomorphism, so in Figure 9, $Seq(A) \cong A$. Moreover, we want this extension to be conservative: a mes with $\square = \varnothing$ (i.e. $\mathcal{W} = \mathcal{C}$) should behave as an event structure. In particular, the sequentialisation of an event structure should be preserved by the inclusion of event structures into mes. So in Figure 9, $Seq(B)$, where B is seen as a mes, should be the same as $Seq(B)$ where B is seen as an event structure.

We also want (a)synchronous products of mes to correspond to (a)synchronous products of event structures. As previously, we use a coreflection to express this preservation. The right adjoint to the inclusion will be the forgetful functor $\mathcal{U}$. But to talk about the coreflection, we first need to define the category of mes. We will provide justification for each of the 4 conditions on maps of mes (Definition 11).

Firstly, since we want $\mathcal{U}$ to be a functor, then for every map of mes f from A to B, there corresponds a map of event structures $\mathcal{U}(f)$ from $\mathcal{U}(A)$ to $\mathcal{U}(B)$.

Secondly, since we want $\mathcal{U}$ to form a coreflection, for every mes A, we need a map from $\mathcal{U}(A)$ to A which is the identity function on events. This implies that we have an identity-on-events map from the mes with two events $a \mathrel{\lrcorner\!\!\urcorner} b$ to the mes with two events $a \square b$. Since we do not want these two mes to be isomorphic, we do not want any identity-on-events map from $a \square b$ to $a \mathrel{\lrcorner\!\!\urcorner} b$. This leads to the second condition on maps of mes: maps of mes preserve worlds.

Thirdly, considering Figure 9 again, any total map g from $Seq(A)$ to $Seq(B)$ has to send the world $\{a_1, a'_1\}$ to a world, hence necessarily $g(a_1) = g(a'_1)$. It follows that for *Seq* to be a functor, any map f from A to B should respect $f(a) = f(a')$. So we need to forbid $f(a) = b$ and $f(a') = b'$ from forming a map. Which leads to the following restriction: if $a \square a'$, f defined on a and a', and $f(a) \neq f(a')$, then $f(a) \square f(a')$.

Lastly, due to some more subtle problems in the case of partial maps (see the full version), we need the domain of the map to be closed under $\square$.

▶ **Definition 11.** *A* map f *of mes from* E *to* F *is a (possibly partial) function from* $|E|$ *to* $|F|$ *satisfying:*
map of event structures f *is a map from* $\mathcal{U}(E)$ *to* $\mathcal{U}(F)$*;*
world preserving *for every* $w \in \mathcal{W}(E)$, $f(w) \in \mathcal{W}(F)$*;*
$\square$***-preserving*** *for every* $a \square_E b$ *where* f *is defined and* $f(a) \neq f(b)$*, then* $f(a) \square_F f(b)$*; and*
$\square$***-equidefinability*** *for every* $a \square_E b$, f *is defined on both or none.*

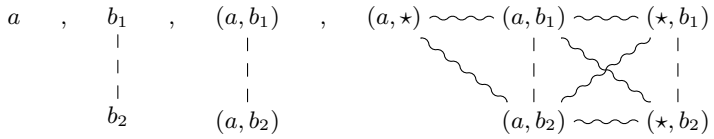


Figure 10 The mes A, B, $A \times B$ and $A \times_\star B$.

▶ **Proposition 12.** *Mes and partial maps of mes form a category, written* $\mathsf{MES}_\star$*, with the empty event structure as a terminal object. Mes and total maps of mes form a subcategory, written* MES*.*

▶ **Theorem 13.** *The functor* $\mathcal{U}$ *is a left adjoint to the inclusion, forming a coreflection.*

$$\mathsf{ES} \underset{\mathcal{U}}{\overset{\hookrightarrow}{\underset{\longleftarrow}{\overset{\longrightarrow}{\bot}}}} \mathsf{MES} \qquad \mathsf{ES}_\star \underset{\mathcal{U}}{\overset{\hookrightarrow}{\underset{\longleftarrow}{\overset{\longrightarrow}{\bot}}}} \mathsf{MES}_\star$$

Additionally, the inclusion preserves categorical limits.

See the full version for the proof. This theorem implies that both the forgetful functor $\mathcal{U}$ and the inclusion functor $\hookrightarrow$ preserve (a)synchronous products.

3.2 Products and Sequentiality

From the previous coreflection, we know that mes coming from event structures have synchronous and asynchronous products. These constructions extend to all mes. The proof is technical, and can be found in the full version.

▶ **Proposition 14.** $\mathsf{MES}_\star$ *is a cartesian category, with* $\times_\star$ *as a product and* $\varnothing$ *as a unit.* MES *has a product* $\times$*.*

As a remark, since we have $\mathcal{U}(A \times_\star B) = \mathcal{U}(A) \times_\star \mathcal{U}(B)$, we know that the events, order, and conflict of the mes $A \times_\star B$ are the same as those of the event structure $\mathcal{U}(A) \times_\star \mathcal{U}(B)$. And we have a similar property for $A \times B$. One can then characterise the internal conflict:

▶ **Proposition 15.** *For* E_1 *and* E_2 *two mes, with* $P = E_1 \times E_2$ *and* $P_\star = E_1 \times_\star E_2$*, we have:*

$$a \mathrel{\square_P} b \iff [a] \cup [b), [a) \cup [b] \in \mathcal{W}(P) \text{ and } \forall i \in \{1,2\}, \begin{cases} \pi_i(a) = \pi_i(b), \text{ or} \\ \pi_i(a) \mathrel{\square_{E_i}} \pi_i(b) \end{cases}$$

$$a \mathrel{\square_{P_\star}} b \iff [a] \cup [b), [a) \cup [b] \in \mathcal{W}(P_\star) \text{ and } \forall i \in \{1,2\}, \begin{cases} \pi_i^\star \text{ undef. on } \{a,b\}, \text{ or} \\ \pi_i^\star(a) = \pi_i^\star(b), \text{ or} \\ \pi_i^\star(a) \mathrel{\square_{E_i}} \pi_i^\star(b) \end{cases}$$

This characterisation is recursive, since it refers to worlds and worlds are defined by the internal conflict. However, this is a well-founded recursion[4], so this proposition could be used as a definition of the (a)synchronous product. Figure 10 is an example of the synchronous and asynchronous product of two mes.

We now aim to extend the definition of sequentiality to mes. However, a problem arises: the usual definition of sequentiality "$\forall a, b \in c \in \mathcal{C}(E) \implies a \leq b$ or $b \leq a$" gives only a many-to-one correspondence with trees, and is no longer equivalent to the categorical

[4] The well-founded relation is $(a', b') \prec (a, b) \iff a' \leq_{E_1 \times_\star E_2} a$ and $b' \leq_{E_1 \times_\star E_2} b$ and $(a', b') \neq (a, b)$.

definition "there exists a total map from E to $\mathcal{N}$". Indeed, since maps are expected to be $\square$-preserving, the mes E_2 from Figure 8 satisfies the first but not the second. So we strengthen the first definition so that it is equivalent to the second: we require internal conflicts to be minimal, i.e. $\square \subseteq \frown$. This allow to recover a one-to-one correspondence with alternating trees (Proposition 20).

▶ **Definition 16.** *A mes E is* sequential *if it satisfies one of the following equivalent properties:*

- *E is forest-shaped, with branches being in conflict with each other, and $\square_E \subseteq \frown_E$.*
- *For every $a, b \in x \in \mathcal{C}(E)$, either $a \leq_E b$ or $b \leq_E a$. Moreover, $\square_E \subseteq \frown_E$.*
- *There exists a (necessarily unique) total map from E to $\mathcal{N}$.*
- *E is isomorphic to $E \times \mathcal{N}$.*

Analogously to the correspondence between trees and event structures, *alternating trees* (Definition 18) will correspond to sequential mes.

▶ **Proposition 17.** *We write* Seq-MES *(resp.* Seq-MES$_\star$*) for the subcategory of* MES *(resp.* MES$_\star$*) with only sequential mes. It is a cartesian category, with $\times$ (resp. $\times_\star$) as a product and $\mathcal{N}$ (resp. $\varnothing$) as a terminal object.*

Now, we define the sequentialisation as previously: for a mes E, we define the sequential mes $Seq(E)$ as $E \times \mathcal{N}$. The coreflection (Theorem 6) still holds in the case of mes. A proof can be found in the full version.

3.3 Labelled Mixed Event Structures

As in the case of event structures, one can add labels to mes. A *mes labelled in $\mathcal{L}$* is a mes E together with a labelling function $\ell_E : |E| \to \mathcal{L}$. Maps are required to preserve labels, Seq preserves labels, and we compute the labels of asynchronous product using $\bullet$.

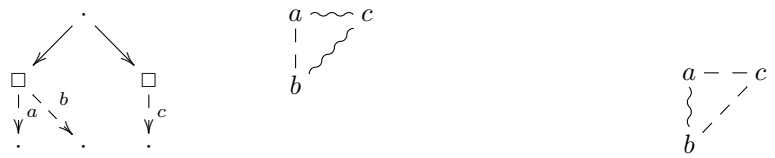
3.4 Alternating Trees

Just as we can choose a root node of an LTS and unfold it into a tree, so can we choose a root node in a Segala automaton and unfold it into a Segala tree. We define here *labelled alternating trees* which can be understood as "Segala trees, but without probabilities"; the complete definition of a Segala tree will come later (Definition 24), and we will omit the definition of Segala automaton [13, 14].

▶ **Definition 18.** *A* labelled alternating tree *is a tree such that:*

- *Nodes of even depth (including the root) are called* states*. All leaves must be states.*
- *Nodes of odd depth are called* intermediary positions*.*
- *Transitions from intermediary positions to states are labelled. They are called* internal transitions *and written $\dashrightarrow_\ell$ if labelled by ℓ.*
- *Transitions from states to intermediary positions are unlabelled. They are called* external transitions *and written $\to$.*

Labelled alternating trees can be represented directly by labelled sequential mes. Figure 11a shows an alternating tree and its representation by a labelled sequential mes. Every intermediary position of the labelled alternating tree corresponds to a *cell* (set of events that are pairwise related by $\square$), and each labelled transition from this intermediary position corresponds to a labelled event of this cell. However, this is not a one-to-one correspondence, since labelled alternating trees will only generate mes where $\square$ is a transitive relation. In particular, the labelled sequential mes in Figure 11b cannot be built using labelled alternating trees.



(a) A alternating tree and its sequential mes. **(b)** A sequential mes corresponding to no alternating tree.

Figure 11 Alternating trees and mes.

▶ **Definition 19.** *A mes E is said to be* locally transitive *if for every $x \overset{a}{\text{—}\!\subset}$, $x \overset{b}{\text{—}\!\subset}$, and $x \overset{c}{\text{—}\!\subset}$ with $x \in \mathcal{C}(E)$ and a, b, c pairwise distinct, we have $a \,\square_E\, b \,\square_E\, c \Rightarrow a \,\square_E\, c$.*

We use this more restricted notion of transitivity because "$\square$ is transitive" is not a property stable under (a)synchronous products, while local transitivity is: the (a)synchronous product of two locally transitive mes is locally transitive. In fact every definition and result previously stated still applies if we restrict ourselves to locally transitive mes.

▶ **Proposition 20.** *There is a one-to-one correspondence between labelled alternating trees and labelled locally transitive sequential mes. Moreover, for a mes E, $Seq(E)$ corresponds to an alternating tree if and only if E is locally transitive.*

4 Concurrent Semantics of Probabilistic CCS

The goal of this section is to give a concurrent semantics to Probabilistic CCS, with a factorisation property similar to the one of CCS. We first add a probabilistic valuation to mes, in order to relate them to Segala trees, used for the interleaving semantics of PCCS. Then, we describe how to extend the concurrent semantics for CCS into one for PCCS.

4.1 Mixed Probabilisitic Event Structures

We recall some notions of probabilistic event structures [15, 20]. They are event structures together with a probability valuation $v : \mathcal{C}(E) \to (0, 1]$, interpreted as *the probability of reaching at least this configuration*, such that $v(\varnothing) = 1$ and a condition of *monotonicity* (Definition 21). Simple consequences of the condition of *monotonicity* are:

- The valuation v is decreasing: $x \subseteq y \implies v(x) \geq v(y)$
- Events in conflict have conditional probability whose sum is less than or equal to one: $\forall 1 \leq i \leq n, x \overset{a_i}{\text{—}\!\subset}$ and $\forall 1 \leq i < j \leq n, a_i \# a_j \implies \sum_{i=1}^{n} \frac{v(x \cup \{a_i\})}{v(x)} \leq 1$
- Events not in conflict respect a variant of the inclusion-exclusion principle: $x, y, x \cap y, x \cup y \in \mathcal{C}(E) \implies v(x \cap y) - v(x) - v(y) + v(x \cup y) \geq 0$

We want to add similar conditions to mes. We consider a mes with two events a and b. We aim to use the internal conflict $\square$ of mes for probabilistic choices $+$ of PCCS, meaning that if $a \,\square\, b$, we can expect $v(\{a\}) + v(\{b\}) \leq 1$, so they respect the *monotone* condition. However, since we aim to use the external conflict $\lightning$ for the nondeterministic choice $\oslash$ of PCCS, if $a \lightning b$, we can have $v(\{a\}) = 1 = v(\{b\})$, which does not respect the *monotone* condition. This guides us to the following condition: within worlds, conflicts are necessarily $\square$, so the *monotone* condition has to be respected, however no condition is expected to hold across different worlds.

▶ **Definition 21.** *A* mixed probabilisitic event structure *(mpes) is a mes E together with a probabilistic valuation $v_E : \mathcal{C}(E) \to (0, 1]$ such that:*

Initialised $v_E(\varnothing) = 1$
Monotone *For* $x, y_1, \dots, y_n \in \mathcal{C}(E)$*, with* $\forall 1 \leq i \leq n, x \subseteq y_i$*, we write:*
- *For* $\varnothing \neq I \subseteq \{1, \dots, n\}, y_I := \bigcup_{i \in I} y_i$
- *For* $X \notin \mathcal{C}(E), v_E(X) := 0$
- $d(x; y_1, \dots, y_n) := v_E(x) - \sum_{\varnothing \neq I \subseteq \{1,\dots,n\}} (-1)^{|I|+1} v_E(y_I)$

We then demand that $y_{\{1,\dots,n\}} \in \mathcal{W}(E) \implies d(x; y_1, \dots, y_n) \geq 0$

It is labelled *if the underlying mes is labelled. It is* sequential *if the underlying mes is sequential. A map of mpes is a map of mes* $f : A \rightharpoonup B$ *such that* $\forall x \in \mathcal{C}(A), v_A(x) \leq v_B(f(x))$.

Unlike [20], we use a valuation that is strictly positive on configurations instead of positive or null. This choice come from the absence of 0-probability branches in Segala trees.

We write $\mathsf{MPES}_\star$, MPES, $\mathsf{Seq\text{-}MPES}_\star$, and $\mathsf{Seq\text{-}MPES}$ for the categories of mpes with maps restricted to total ones and/or with objects restricted to sequential ones. The empty mpes $\varnothing$ is a terminal object for $\mathsf{MPES}_\star$ and $\mathsf{Seq\text{-}MPES}_\star$, while $\mathcal{N} = (\mathbb{N}, \leq, \varnothing, \varnothing, v_{\mathcal{N}} : x \mapsto 1)$ is a terminal object for $\mathsf{Seq\text{-}MPES}$.

We now want to extend the (a)synchronous product to mpes, but it is unfortunately not always possible to preserve the universal property of (a)synchronous products.

▶ **Proposition 22.** MPES *and* $\mathsf{MPES}_\star$ *do not have all products.*

A counterexample can be found in the full version. While no categorical product can be defined, both MPES and $\mathsf{MPES}_\star$ have symmetric monoidal products $\otimes$ and $\otimes_\star$ which generalise $\times$ and $\times_\star$ of mes. The underlying problem of the absence of products is a problem of *probabilistic correlation*. We fail to define $A \times B$ because the correlations between the events of A and the events of B is unspecified. So when defining $A \otimes B$, we make the most canonical choice and consider that A and B are probabilistically independent. This choice is also consistent with the processes of PCCS: when considering $(P|Q)$, the probabilistic choices inside P are assumed to be independent of the probabilistic choices inside Q.

▶ **Definition 23.** *For* A *and* B *two mpes, we define* $A \otimes_\star B$ *as follows:*
- *The underlying mes is* $A \times_\star B$*, where* A *and* B *are seen as mes.*
- $\forall x \in \mathcal{C}(A \otimes_\star B), v_{A \otimes_\star B}(x) = v_A(\pi_1^\star(x)) \cdot v_B(\pi_2^\star(x))$.

This is a mpes (proof in the full version). We define $A \otimes B$ *similarly.*

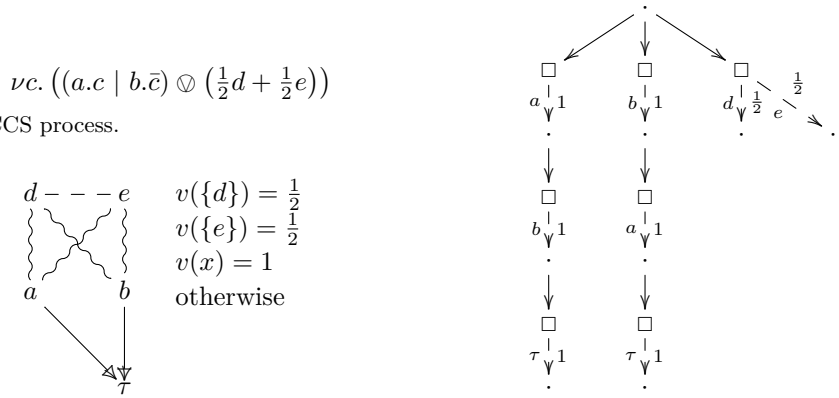
A remarkable property is that when A (or B) respects $\forall x \in \mathcal{C}(A), v_A(x) = 1$, then $A \otimes_\star B$ is a product in $\mathsf{MPES}_\star$, and $A \otimes B$ too in MPES. The sequentialisation $Seq(E) := E \otimes \mathcal{N}$ still induces a coreflection.

We stated earlier that alternating trees were "Segala trees without probabilities", and we will now define Segala trees, and explain their correspondence with locally transitive sequential mpes.

▶ **Definition 24.** *A Segala tree labelled by* $\mathcal{L}$ *is an* alternating tree *labelled by* $(0, 1] \times \mathcal{L}$*, such that if we have* $i \dashrightarrow_{(p_k, a_k)} s_k$ *for* $1 \leq k \leq n$ *(with* s_k *being distinct), then* $\sum_{k=1}^n p_k \leq 1$.

▶ **Theorem 25.** *There is a one-to-one correspondence between Segala trees and labelled locally transitive sequential mpes.*

This correspondence is given by the correspondence between labelled alternating trees and labelled locally transitive sequential mes, with the probability of a transition $\dashrightarrow_{(p,a)}$ corresponding to an event e given by $p = \frac{v([e])}{v([e])}$, and reciprocally the valuation of a configuration x being the product of all the probabilities on transitions of the Segala tree along the branch corresponding to x.



(a) A PCCS process.

(b) Its concurrent semantics: a labelled mixed probabilisitic event structure.

(c) Its interleaving semantics: a Segala tree.

Figure 12

4.2 Concurrent Semantics of PCCS

In order to define the concurrent semantics of PCCS, we first extend the constructors on event structures used for CCS to mpes.

- $\mathcal{U}(E\backslash L) = \mathcal{U}(E)\backslash L$; $\Box_{E\backslash L}$ and $v_{E\backslash L}$ are the restriction of $\Box_E$ and v_E to $|E\backslash L|$.
- $\mathcal{U}(A \# B) = \mathcal{U}(A) \# \mathcal{U}(B)$; $\Box_{A \# B}$ is the disjoint union, and $v_{A \# B}$ is the co-pairing.
- $\mathcal{U}(\downarrow_{e:a} E) = \downarrow_{e:a} \mathcal{U}(E)$; $\Box_{\downarrow_{e:a}E}$ and $v_{\downarrow_{e:a}E}$ are the extension of $\Box_E$ and v_E to $|\downarrow_{e:a} E|$ (so in particular $v_{\downarrow_{e:a}E}(\{e\}) = v_E(\varnothing) = 1$)
- Recursion is still a least fixpoint for the "substructure maps", i.e. total maps that are an inclusion on events, and preserve and reflect order, internal and external conflicts, and the valuation.

This allows us to represent every process of CCS as an mpes, using the same notation as in Figure 5b (with $\otimes$ instead of $\times$). We define the operation $+$ on mpes as follows. For $(p_i)_{i\in I} \in (0,1]^I$ and $\sum_{i\in I} p_i \leq 1$, we define $S = \sum_{i\in I} p_i \cdot E_i$ with:

- $\mathcal{U}(S) = \#_{i\in I}\mathcal{U}(E_i)$ and $v_S(x) = p_i \cdot v_{E_i}(x)$ if $x \in \mathcal{C}(E_i)$
- $e \,\Box_S\, e' \iff \exists i \in I, \begin{cases} e \,\Box_{E_i}\, e', \text{ and} \\ e, e' \in |E_i| \end{cases}$ or $\exists i \neq j \in I, \begin{cases} e \in |E_i|, \text{ and} \\ e' \in |E_j| \end{cases}$

The concurrent semantics $[\![\_]\!]_{\text{cs}}$ of PCCS is given by Figure 5b together with this additional rule: $[\![\sum_{i\in I} p_i \cdot a_i.P_i]\!]_{\text{cs}} := \sum_{i\in I} p_i \cdot [\![a_i.P_i]\!]_{\text{cs}}$. See Figure 12 for an example.

We note that for every process P of PCCS, $[\![P]\!]_{\text{cs}}$ is locally transitive, meaning that its sequentialisation will correspond to a Segala tree. If we write $[\![P]\!]_{\text{is}}$ for the semantics of the process P with Segala trees, seen as labelled, locally transitive, sequential mpes, then we have the following factorisation theorem.

▶ **Theorem 26** (Factorisation). *For any process P of PCCS, we have $[\![P]\!]_{is} \cong Seq([\![P]\!]_{cs})$.*

The proof of this theorem relies on the sequentialisation coreflection, which ensures that the interpretation of the parallel composition is preserved.

This concludes the main contribution of this paper: we have built a concurrent model able to represent both probabilistic and nondeterministic choices, and used it to give to PCCS a concurrent semantics compatible with its usual interleaving semantics.

As future work, we wish to investigate PCCS without the guard restriction, hence allowing *unguarded probabilistic choice* $\sum_{i \in I} p_i \cdot P_i$. We are able to give to Unguarded PCCS a concurrent semantics using potentially non locally transitive mpes, however the relationship with existing interleaving semantics remains unclear.

References

1 Jos C. M. Baeten. A brief history of process algebra. *Theor. Comput. Sci.*, 335(2-3):131–146, 2005. doi:10.1016/j.tcs.2004.07.036.

2 Christel Baier and Marta Z. Kwiatkowska. Domain equations for probabilistic processes. *Electr. Notes Theor. Comput. Sci.*, 7:34–54, 1997. doi:10.1016/S1571-0661(05)80465-7.

3 Jan A. Bergstra and Jan Willem Klop. Process Algebra for Synchronous Communication. *Information and Control*, 60(1-3):109–137, 1984. doi:10.1016/S0019-9958(84)80025-X.

4 Simon Castellan. Weak memory models using event structures. In Julien Signoles, editor, *Vingt-septièmes Journées Francophones des Langages Applicatifs (JFLA 2016)*, Saint-Malo, France, January 2016. URL: https://hal.inria.fr/hal-01333582.

5 Simon Castellan, Pierre Clairambault, Hugo Paquet, and Glynn Winskel. The Concurrent Game Semantics of Probabilistic PCF. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '18, pages 215–224, New York, NY, USA, 2018. ACM. doi:10.1145/3209108.3209187.

6 Pierre Clairambault, Marc de Visme, and Glynn Winskel. Game semantics for quantum programming. *PACMPL*, 3(POPL):32:1–32:29, 2019. URL: https://dl.acm.org/citation.cfm?id=3290345, doi:10.1145/3290345.

7 Jean Goubault-Larrecq. Isomorphism theorems between models of mixed choice. *Mathematical Structures in Computer Science*, 27(6):1032–1067, 2017. doi:10.1017/S0960129515000547.

8 Oltea Mihaela Herescu and Catuscia Palamidessi. Probabilistic asynchronous pi-calculus. *CoRR*, cs.PL/0109002, 2001. arXiv:cs.PL/0109002.

9 Gilles Kahn, editor. *Semantics of Concurrent Computation, Proceedings of the International Symposium, Evian, France, July 2-4, 1979*, volume 70 of *Lecture Notes in Computer Science*. Springer, 1979. doi:10.1007/BFb0022459.

10 Robin Milner. Lectures on a Calculus for Communicating Systems. In *Seminar on Concurrency, Carnegie-Mellon University, Pittsburg, PA, USA, July 9-11, 1984*, pages 197–220, 1984. doi:10.1007/3-540-15670-4_10.

11 Mogens Nielsen, Gordon D. Plotkin, and Glynn Winskel. Petri Nets, Event Structures and Domains. In Gilles Kahn, editor, *Semantics of Concurrent Computation, Proceedings of the International Symposium, Evian, France, July 2-4, 1979*, volume 70 of *Lecture Notes in Computer Science*, pages 266–284. Springer, 1979. doi:10.1007/BFb0022474.

12 Mogens Nielsen and Erik Meineche Schmidt, editors. *Automata, Languages and Programming, 9th Colloquium, Aarhus, Denmark, July 12-16, 1982, Proceedings*, volume 140 of *Lecture Notes in Computer Science*. Springer, 1982. doi:10.1007/BFb0012751.

13 Roberto Segala. *Modeling and verification of randomized distributed real-time systems*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1995. URL: http://hdl.handle.net/1721.1/36560.

14 Ana Sokolova and Erik P. de Vink. Probabilistic Automata: System Types, Parallel Composition and Comparison. In *Validation of Stochastic Systems - A Guide to Current Research*, pages 1–43, 2004. doi:10.1007/978-3-540-24611-4_1.

15 Daniele Varacca, Hagen Völzer, and Glynn Winskel. Probabilistic event structures and domains. *Theor. Comput. Sci.*, 358(2-3):173–199, 2006. doi:10.1016/j.tcs.2006.01.015.

16 Daniele Varacca and Nobuko Yoshida. Probabilistic pi-Calculus and Event Structures. *Electr. Notes Theor. Comput. Sci.*, 190(3):147–166, 2007. doi:10.1016/j.entcs.2007.07.009.

17 Glynn Winskel. Event Structure Semantics for CCS and Related Languages. In Mogens Nielsen and Erik Meineche Schmidt, editors, *Automata, Languages and Programming, 9th Colloquium, Aarhus, Denmark, July 12-16, 1982, Proceedings*, volume 140 of *Lecture Notes in Computer Science*, pages 561–576. Springer, 1982. doi:10.1007/BFb0012800.

18 Glynn Winskel. Event Structures. In *Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986, Part II, Proceedings of an Advanced Course, Bad Honnef, Germany, 8-19 September 1986*, pages 325–392, 1986. doi:10.1007/3-540-17906-2_31.

19 Glynn Winskel. Event Structures with Symmetry. *Electr. Notes Theor. Comput. Sci.*, 172:611–652, 2007. doi:10.1016/j.entcs.2007.02.022.

20 Glynn Winskel. Probabilistic and Quantum Event Structures. In *Horizons of the Mind. A Tribute to Prakash Panangaden - Essays Dedicated to Prakash Panangaden on the Occasion of His 60th Birthday*, pages 476–497, 2014. doi:10.1007/978-3-319-06880-0_25.